Evaluation of Safe Control Policies for Mobile Robots in Dynamic Environments

Varun Murali, Niharika Arora, Ian Buckley

***Abstract*—With the growth of domestic robot industry, it is important to understand the navigation of mobile robots in dynamic environments. Use of mobile robots in the home can greatly improve the quality of life and is a growing field of research. With an increased interaction between humans and robots, and an increasingly shared workspace, it is important to study the safety of the robotic system and the extent of guarantees that can be made to minimize risk of injury. We present a nonlinear control approach to the navigation problem that achieves the objective of reaching pre-defined locations with provable guarantess of safety. Verification of the controller is performed first in MATLAB simulation, which shows that the controller is capable of driving the system to a pre-defined location while mainting all required safety rules defined in it's environment. The system is further investigated on the robot simulator, Gazebo and implemented on a Segway RMP 200 based robot, demonstrating that the nonlinear navigation controller successfully traverses the gap between theory and practice.**

I. Introduction

Navigation is of fundamental concern for mobile robots. The ability to successfully avoid obstacles, both static and dynamic, while moving towards a goal location largely determines the utility of a mobile robotic platform. It is critical that robots avoid obstacles for their own safety as well as the safety of the people around them. Given the importance of navigation, it is not surprising that the problem has long been considered and explored; furthermore, it is reasonable to assert that approaches to the problem are as varied as the disciplines related to robotics themselves.

Both deliberative and reactive approaches to navigation have been explored. Deliberative approaches navigation benefit from optimal path planning, but suffer when reality deviates from the map, either due to mapping inaccuracies or dynamic obstacles. Probabilistic roadmaps for path planning, first presented by Kavraki et. al. in [1], are a deliberative approach to navigation in which a search algorithm is applied to a map in order to identify a possible path through the environment. In contrast, reactive approaches to navigation may not find optimal solutions and are susceptible to local minima but are able to respond to dynamic obstacles and the environment as it is encountered. Artificial potential fields are a reactive method of navigation that have long been considered [2]. Hybrid approaches to navigation take planning and layer it with reactive controls that handle possible obstacles.

Nonlinear control based navigation is also a studied problem such as by Ting et al. [3]. Generating a policy in a reactive fashion reduces the complexity of the computation needed. Traditional path planners can be used to provide a high level trajectory by having some previous metric knowledge of the space. Lower level reactive controllers can be used for local refinement of the global path and can be used to control the robot–this reduces the complexity of re-planning every time a change in the environment is observed. Potential field based algorithms have been used to generate such reactive policies in the past with some success. There are some known pitfalls for this method with regards to local minima but several solutions for this have been proposed.

Given the possibility of having a global metric representation of a domestic environment, a global trajectory can be generated and a local non-linear controller that obeys constraints imposed by the robot and the objective could be used to achieve navigational objectives. In this paper, control Lyapunov and barrier functions are designed for robot navigation and collision avoidance. Constraints important for navigation, such as distance from obstacles, distance to target, deviation from desired heading, minimum and maximum velocity, and minimum and maximum rotational velocity will be combined to generate a navigation control protocol for the robot. Quadratic programming as presented by Ames et. al. in [4] will be used to simultaneously satisfy these constraints. Furthermore, by applying nonlinear control to navigation, claims about stabilization of the robot to the navigational directive can be guaranteed theoretically, which is an improvement over other reactive navigation methods, which can make no such claims.

The navigation control protocol will first be motivated and derived. After explaining the protocol, MATLAB simulation will be used to verify that the navigation control protocol correctly drives the state of the robot to a target. After demonstrating that the robot can navigate to a target location, a trajectory will be generated for a map using a probabilistic roadmap, and the control law will be used to drive the robot along the trajectory. After thoroughly verifying the navigation control protocol in simulation, the controller will be implemented in the ROS framework and evaluated on the mobile robot Jeeves, shown in Figure 1, which will verify the navigation control protocol in a real world scenario.

II. System Description and Dynamics

The robotic system considered in this paper and its dynamics are presented. The dynamics are simplified to allow design of a navigation controller with safety constraints. The following sections describe the system and detail the dynamics.

A. System Description

The robot platform shown in Figure 1 consists of a Segway RMP-200 mobile base, which has been modified to hold

Fig. 1: Jeeves - A modified Segway based robot mounted with a Primesense, Microsoft Kinect, Roboteye RE05, Hokuyo LTM-30x laser and the UR5. Currently modified to include the casters for static stability

various sensors. The Seqway RMP 200 has been modified to be statically stable. The sensor suite on the robot consists of a Roboteye RE05 sensor capable of scanning in 3D, a primesense RGB-D camera and a Kinect RGB-D. The Hokuyo LTM-30x is mounted at the base and used for base obstacle avoidance. A microstrain IMU is mounted on the robot to estimate the state of the robot.

In the scope of this project, the Hokuyo LTM-30x is used to detect obstacles from a planar scan. The closest laser hit can be considered an obstacle and dynamic obstacles can be modelled in this way. The robot is controlled using a PS3 controller when in tele-operation mode, which has a manual override and a dead-man switch to kill the robot due to any type of failure. The onboard computer on the robot is an i7 computer with 16 GB of RAM.

B. Nonlinear Dynamics

The robot can be modelled by unicycle dynamics for the purpose of designing the navigation control protocol proposed by this paper. The unicycle dynamics model is presented in the familiar rectangular form:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} v\ \cos(\theta) \\ v\ \sin(\theta) \\ \omega \end{pmatrix}, \tag{1}$$

where (x, y) is the 2D position of the robot, and $\theta \in (-\pi, \pi]$ is the angle formed between the heading of the robot and the positive x-axis; v and w are the translational and angular velocities respectively. Park et al [5] showed the possibility of using polar co-ordinates to generate a smooth controller. They also propose that designing a controller in polar co-orodinates leads to more "human like" motion. In polar co-ordinates the dynamics can be written in the form of r, θ, and δ. Figure 2 describes these states. The r is the radial distance from the goal, the θ is the orientation of the target reference frame in the robot frame, δ is the angle between the heading of the robot and the vector joining the center of the robot and the target location. To simplify the problems, a change of co-ordinates is used to position the target location at the origin. The final form of the dynamics is given by:

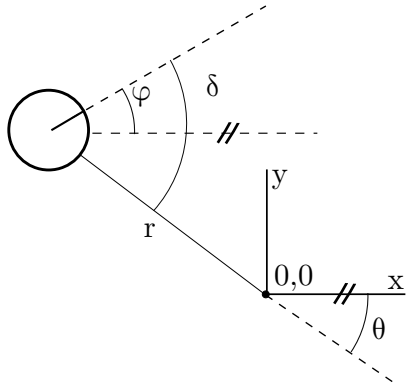


Fig. 2: The polar co-ordinates used to design the controller are shown in the figure above. The r is the radial distance of the robot from the origin, the angle ϕ is the heading of the robot in it's own reference frame, θ is the orientation of the target frame in the robot frame, and δ is the angle between the heading and the vector joining the robot center and the origin.

$$\begin{pmatrix} \dot{r} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} -v\ \cos(\delta) \\ \frac{v}{r}\ \sin(\delta) \end{pmatrix}. \tag{2}$$

$$\dot{\delta} = \frac{v}{r}\ \sin(\delta) + \omega \tag{3}$$

The dynamics of the robot can be represented in affine form ($\dot{x} = f(x) + g(x)u$), where

$$u = \begin{pmatrix} v \\ \omega \end{pmatrix}, \tag{4}$$

and it is obvious that the drift term is zero.

C. Control Objectives

The objective of this project is to develop a nonlinear control policy for navigation of mobile robots. To do this appropriate control objectives must be identified and constraints on the inputs must be determined; these constraints will be used to generate a navigation controller for the Segway robot to ensure that the constraints are obeyed. The stability of the system with respect to the Lyapunov-like barrier functions can then be analysed, which provides theoretical guarantees of the controller performance in achieving the high level objectives for the robot. The experimental objective is to implement the controller on a real robot to study the gap between theory and practice.

Control of unicycle robots has been explored. The objective in [5] was to drive a wheelchair robot to a desired point and heading in a manner similar to the way that humans intuitively

drive. To do this, positive linear velocities ($v > 0$) are enforced, which prevents the wheelchair robot from going backwards in a direction that its passenger cannot see; this approach is useful in a broader context, most relevant is its application to robots with rigidly mounted, forward facing sensors such as the robot considered by this paper.

Hard Constraints: The most important constraint on the system is that the robot not collide with obstacles. To achieve this objective, the distance to the obstacle must not be smaller than a minimum safe distance, which can be represented in the following expression:

$$z_b > z_{safe}. \tag{5}$$

Soft Constraints: The objective of the controller is to drive the state of the robot to zero i.e:

$$\dot{x} = (\dot{r}, \dot{\theta}, \dot{\delta})^T \rightarrow (0, 0, 0) \tag{6}$$

. In the context of the real system, this will mean asymptotically driving the distance from the target position to zero while aligning the heading with zero radians in the global coordinate frame. Furthermore, driving the system to zero exponentially is preferred.A soft constraint on the obstacle avoidance is also added to ensure that the robot can exit a dangerous situation if it enters a region where it is not yet considered unsafe but is approaching the possibility of a dangerous situation.

Actuator Constraints: Constraints on the maximum linear and angular velocity inputs can be applied to restrict the controller to physically achievable actuator effort i.e:

$$c_a <= v <= c_b \tag{7}$$

$$c_c <= w <= c_d \tag{8}$$

where v is the translational velocity, w is the angular velocity, c_a and c_b are the minmum and maximum translational velocity that can be actuated and c_c and c_d are the minimum and maximum angular velocity that are physically feasible.

III. Control Framework

This section details the development of the safe navigation control protocol. In particular, the soft and hard constraints previously presented motivate the use of control Lyapunov functions and barrier functions for achieving safe navigation.

A. Soft Constraints as Control Lyapunov Functions

To design the control law, the standard quadratic Lyapunov function candidate is first considered:

$$V = \frac{1}{2}(r^2 + \theta^2) \tag{9}$$

$$\dot{V} = r\dot{r} + \theta\dot{\theta} \tag{10}$$

$$= -rv\ \cos(\delta) + \theta v\ \sin(\delta) \tag{11}$$

Fixing δ to enforce a virtual steering control as in [5],

$$\delta = \arctan(-k_1\theta). \tag{12}$$

The derivative of the Lyapunov function becomes:

$$\dot{V} = -rv\ \cos(\arctan(-k_1\theta)) + \theta v\ \sin(\arctan(-k_1\theta)). \tag{13}$$

Because $\theta \in (-\pi, \pi]$, $v > 0$, and $r \geq 0$, $\dot{V} < 0\ \forall \theta, r \neq 0$; thus, the steering control asymptotically drives $r, \theta \rightarrow 0$. By choosing $v = k_3 r$ in some neighborhood of $r = 0$ for positive constant k_3, the singularity at $r = 0$ is removed and the system is globally asymptotically stable. Furthermore, by the definition of the steering control, $\delta \rightarrow 0$ as $\theta \rightarrow 0$, so the virtual steering control not only drives the robot to the final position, but aligns the heading of the robot with the positive x-axis of the global coordinate frame by the time it arrives at the final position.

To drive the state to zero through the steering control, the heading must be driven such that $\delta = \arctan(-k_1\theta)$. An objective z_1 is defined:

$$z_1 \equiv \delta - \arctan(-k_1\theta). \tag{14}$$

The derivative is calculated to yield:

$$\dot{z_1} = \left(1 + \frac{k_1}{1 + (k_1\theta)^2}\right)\frac{v}{r}\ \sin(z_1 + \arctan(-k_1\theta)) + \omega. \tag{15}$$

A specific example of a control law that can achieve this objective is shown through feedback linearization by choosing the angular velocity

$$\omega = -\frac{v}{r}\left[k_2 z_1 + \left(1 + \frac{k_1}{1 + (k_1\theta)^2}\right)\sin(z_1 + \arctan(-k_1\theta))\right], \tag{16}$$

the objective dynamics are globally exponentially stable with the following form:

$$\dot{z_1} = -k_2\frac{v}{r}z_1, \tag{17}$$

where k_1 and k_2 can be chosen to increase or decrease how aggressive the steering control is. This is modelled along with other constraints by defining:

$$V_2 = \frac{1}{2}z^2 \tag{18}$$

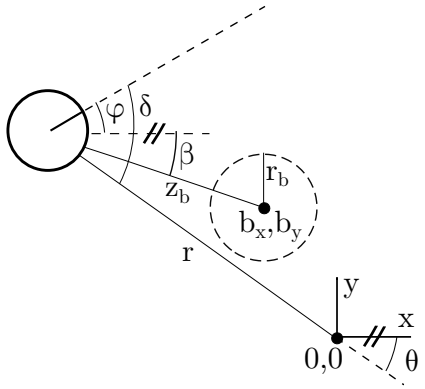


Fig. 3: The figure above shows the egocentric co-ordinate system of the robot. In addition, the obstacle is visulaized. The angle β is the angle the vector joining the center of the robot to the obstacle makes with respect to the robot heading.

A second steering controller is designed to drive the state away from the origin and is defined by the following objective:

$$z_2 = \delta - \text{ atan}(k_1\beta), \tag{19}$$

where β is the angle formed between the global positive x-axis and the vector z_b pointing from the robot to the obstacle. Figure 3 shows the egocentric coordinate system with the addition of the obstacle, where b_x, b_y is the global coordinate of the obstacle and r_b is the minimum safe distance.

The derivative of z_2 is similarly computed as with z_1, and is given by

$$\dot{z_2} = \dot{\delta} \tag{20}$$

By choosing the angular velocity input w to drive the heading $\delta = \text{ arctan}(-k_1\theta)$, steering control, which drives the state to zero, is achieved. Similarly, choosing the angular velocity input w to drive the heading $\delta = \text{ arctan}(k_1\beta)$ causes the robot to drive away from zero. Thus, there exists a control input that satisfies the control Lyapunov function $\dot{V} < 0$; furthermore, following Definition 3 in [6], the control Lyapunov function is exponentially stabilizing if it satisfies

$$\inf_{u \in U} \left[\dot{V} + \gamma V\right] \leq 0 \tag{21}$$

for some constant $\gamma > 0$, so choosing the correct inputs u will achieve the desired exponential stabilization. According to [4], by picking a soft constraint relaxation variable λ, the condition for exponential stabilization shown in (21) is relaxed and can be enforced simultaneously along with a single hard constraint. Because control of the robotic system requires either $\dot{z_1} = -k_2\frac{v}{r}z_1$ or $\dot{z_2} = -k_2\frac{v}{r}z_2$ and $\dot{V} < 0$, two soft constrained expressions must be satisfied at a time to achieve exponential stabilization of the state to zero.

B. Hard Constraints as Barrier Functions

As in [4] and [6], control barrier functions are a logical way of enforcing safety constraints in navigation from a nonlinear control perspective.

A zeroing barrier function is used to drive the robot away from obstacles if it leaves the safe set (e.g. the robot gets within an unsafe threshold distance of the obstacle). The safe set C is described by the following expressions:

$$C = x \in \mathbb{R}^3 | z_b \geq z_{safe} \tag{22}$$

$$\partial C = x \in \mathbb{R}^3 | z_b = z_{safe} \tag{23}$$

$$\text{Int}(C) = x \in \mathbb{R}^3 | z_b > z_{safe} \tag{24}$$

As presented in [7], a barrier function is a zeroing barrier function $h(x)$ if $\dot{h}(x) \leq \gamma h(x)$. For the purpose of avoiding obstacles in the event that the robot leaves the safe set, a zeroing barrier function was chosen with the following form:

$$h(x) = \sqrt{(-v\,\cos(\theta) - b_x)^2 + (v\,\sin(\theta) - b_y)^2}. \tag{25}$$

The derivative of $h(x)$ is given by the following expression:

$$\begin{aligned}\dot{h}(x) = \frac{1}{h(x)}[&(-v\,\cos(\delta)(r - b_y\,\sin(\theta) + b_x\,\cos(\theta)) \\ &-v\,\sin(\delta)(b_x\,\sin(\theta) + b_y\,\cos(\theta))]\end{aligned} \tag{26}$$

A barrier function was chosen to prevent the robot from leaving the safe set. According to Definition 2 in [4], the barrier function B is a control barrier function if

$$\inf_{u \in U} \left[\dot{B} + \frac{\gamma}{B}\right] \leq 0. \tag{27}$$

Thus, by satisfying the equation $\dot{B} \leq 1/B$, the barrier function will prevent the robot from leaving the safe set. The barrier function was chosen to be

$$B = \frac{1}{h(x) - z_{safe}}, \tag{28}$$

and its derivative is given by

$$\dot{B} = \frac{-\dot{h}(x)}{(h(x) - z_{safe})^2}. \tag{29}$$

C. QP Based Controller

Quadratic programming is used to find control inputs that simultaneously satisfy both the soft and hard constraints of the system. To minimize control inputs, the cost function $\mathbf{u}^T H\mathbf{u} + F^T\mathbf{u}$ is chosen. The QP is given by the following expression:

$$u^*(x,z) = \underset{\mathbf{u}=\begin{bmatrix} v \\ \omega \\ \lambda_1 \\ \lambda_2 \\ \alpha_1 \\ \alpha_2 \end{bmatrix} \in \mathbb{R}^6}{\arg\min} \quad \frac{1}{2}\mathbf{u}^T H\mathbf{u} + F^T\mathbf{u} \tag{30}$$

$$\text{s.t.} \qquad \dot{V_1} - \alpha_1 V_1 \leq \lambda_1 \tag{31}$$

$$\dot{V_2} - \alpha_2 V_2 \leq \lambda_2 \tag{32}$$

$$\dot{h(x)} - \alpha h(x) \leq \lambda_1 \tag{33}$$

$$\dot{B} \leq 1/B \tag{34}$$

$$0 \leq v \leq 2.0 \tag{35}$$

$$-0.1 \leq \omega \leq 0.1 \tag{36}$$

$$\alpha_1, \alpha_2 \leq 0 \tag{37}$$

In (30), the matrix $H \in \mathbb{R}^{6\times 6}$ has the form

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \tag{38}$$

and the vector $F^T \in \mathbb{R}^{1\times 6}$ has the form

$$F^T = [0 \quad 0 \quad 0 \quad 0 \quad 10 \quad 1]. \tag{39}$$

The last two elements in the diagonal of H were chosen to be zero to ensure that the negative quadratic cost of the slack variables α_1 and α_2 is as large as possible. Conversely, the last two elements of F^T were chosen to ensure that α_1 and α_2 were negative to minimize the cost.

The control Lyapunov functions V_1 and V_2 are the Lyapunov functions correspond to the objectives of driving the state to zero and enforcing the steering control respectively; the equations are given by

$$V_1 = \frac{1}{2}(r^2 + \theta^2) \tag{40}$$

and

$$V_2 = \frac{1}{2}z_*^2. \tag{41}$$

z_* in V_2 represents the different steering objectives z_1 and z_2 in (14) and (19). Because there are two steering objectives, two QP control laws are required: one which drives the state towards the target while avoiding obstacles, and the other that drives the state in such a way that the robot steers away from obstacles. Toggling between steering objectives is dependent on the obstacle measurement. If the measurement of the distance between the robot and the obstacle was less than the minimum safe distance, then the steering objective z_2 is used. Otherwise, z_1 is used, and $\alpha_1 = 0$ is additionally enforced. $h(x)$ and B in the QP controller are given in equations (26) and (28).

IV. Results

To test the ability of the QP controller to drive the robot to a target position while simultaneously avoiding obstacles, both simulation and physical experiments were conducted. Simulation results were generated in both MATLAB and Gazebo, and physical results were generated using the robotic platform Jeeves.

A. Simulation Results

Two distinct MATLAB simulations were used to validate and test the QP navigation controller. In the first simulation, the controller demonstrates successful navigation of the robot to the target position with and without the presence of obstacles; plots of the state are shown, and the Lyapunov function and its derivative are tracked numerically. In the second simulation, a probabilistic roadmap was used to generate a target trajectory through a map, and the QP navigation controller was applied to drive the robot to the target trajectory while simultaneously avoiding obstacles.

Gazebo simulation of a unicycle robot navigating via the QP controller was used to demonstrate the feasibility of the controller in practice.

1) Target Position: Figure 4 shows the output of the first simulation. The navigation objective was to drive the robot from the starting position and heading to zero at the origin. The robot was initialized from eight starting locations, shown in red, with random egocentric headings. The blue lines show the trajectory of the robot, and as expected, the QP controller successfully accomplishes the navigation objective.

To further evaluate the QP controller and substantiate its success, the state is plotted. Figures 5 shows that the distance to the target position, the angle θ, and the egocentric heading are successfully zeroed under the QP navigation control protocol.

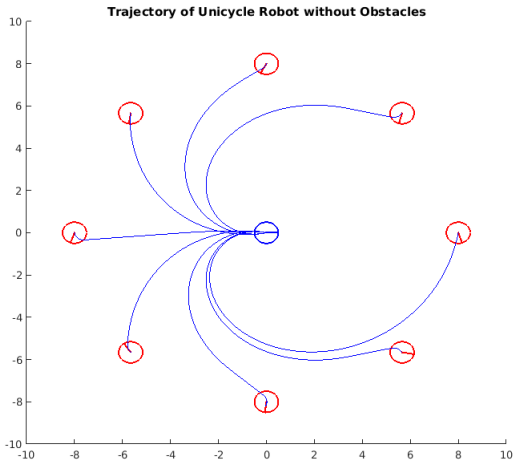


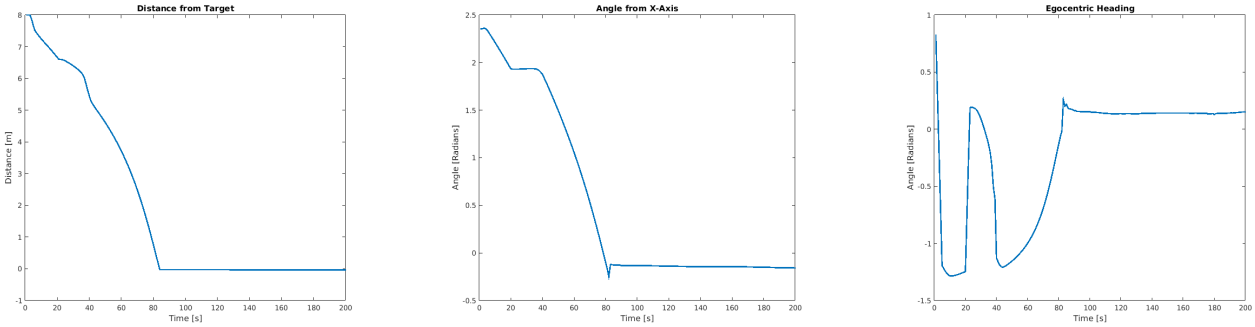
Fig. 4: Robot Trajectory in the Absense of Obstacles

To test the ability of the QP navigation control protocol to prevent collision of the robot with obstacles, robots were again initialized from eight starting positions with random headings, but an obstacle was placed in each robot's path. As Figure 6 demonstrates, the QP navigation controller simultaneously prevented collision while driving the robot to the desired state.

To numerically validate the relaxed control Lypanuov function in the QP, the value of the Lyapunov function and its derivative were plotted as the robot navigated. For the situation in which obstacles are absent, Figure 7 shows the Lyapunov function and its derivative. The derivative is strictly less than zero until the state is driven to zero. The Lyapunov function is strictly positive until the robot reaches its final destination, but it is not strictly decreasing; this is an artifact resulting from the slack variable, which allow the QP to satisfy all of the constraints. Figure 8 shows that the Lyapunov function and its derivative still behave correctly in the presence of obstacles.

2) Target Trajectory: To demonstrate that the QP navigation control protocol enables successful navigation and collision avoidance, the probabilistic roadmap (PRM) planner was used to generate a trajectory through and environment. A path was planned using PRM for the map as shown in Figure **??**. The result of following this trajectory under QP control policy with the barrier functions removed is shown in Figure **??**; clearly, without the barrier functions, the robot fails to arrive safely to its final location. Figure **??** shows the result of applying the QP controller with the barrier functions, and in contrast to the previous case, the robot avoids obstacle and navigates safely.

3) Gazebo: Using Gazebo, the QP navigation control policy was implemented on a unicycle robot in a cluttered environment. Figure **??** shows the trajectory of the robot as it navigates from its starting position to the final location. By demonstrating that the QP controller enabled successful navigation in a simulated 3D environment using the on-board sensors of the unicycle robot, the feasibility of implementing the QP navigation control protocol on a physical system was



(a) Distance of Robot to Target: r (b) Angle between Positive x-axis and r: θ (c) Egocentric Heading of Robot: δ

Fig. 5: Robot State under QP Navigation Protocol

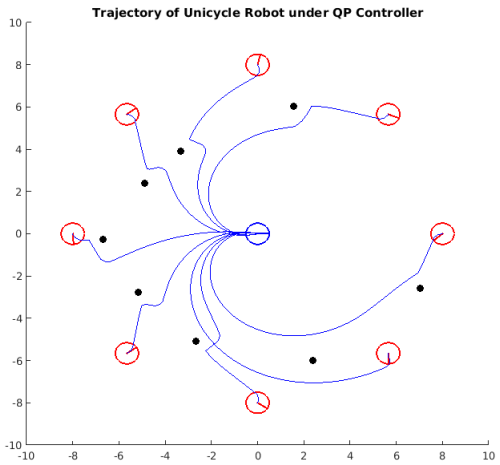


Fig. 6: Robot Trajectory in the Presense of Obstacles

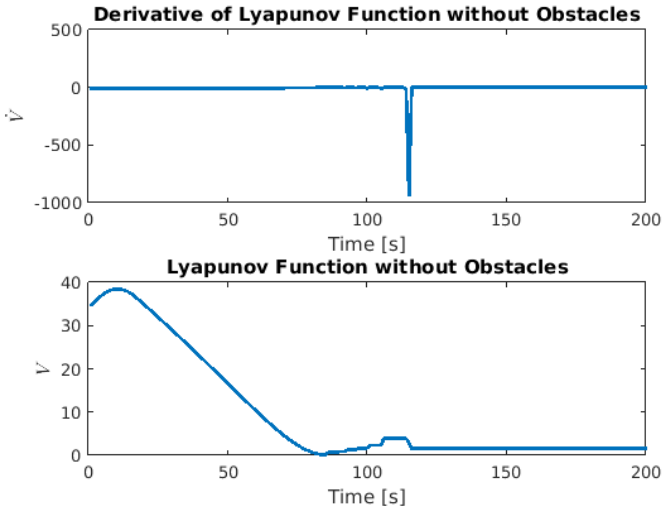


Fig. 7: Lyapunov Function without Obstacles

established.

B. Experimental Results

V. Conclusion

References

[1] L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," in *RIEEE Transactions on Robotics and Automation (Volume:12 , Issue: 4)*. IEEE, 1996, pp. 566–580.

[2] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Robotics and Automation. Proceedings. 1985 IEEE International Conference on (Volume:2)*. IEEE, 1985, pp. 500–505.

[3] L. Ting, C. J. Macnab, and S. Magierowski, "Reactive robot navigation utilizing nonlinear control," *International Journal of Advanced Robotic Systems*, vol. 11, 2014.

[4] A. D. Ames, J. W. Grizzles, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," in *53rd IEEE Conference on Decision and Control*. IEEE, 2014, pp. 6271–6278.

[5] J. J. Park and B. Kuipers, "A smooth control law for graceful motion of differential wheeled mobile robots in 2d environment," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 4896–4902.

[6] A. D. Ames, K. Galloway, K. Sreenath, and J. W. Grizzle, "Rapidly exponentially stabilizing control lyapunov functions and hybrid zero dynamics," in *IEEE Transactions on Automatic Control (Volume:59 , Issue: 4)*. IEEE, 2014, pp. 876–891.

[7] X. Xu, P. Tabuada, J. W. Grizzle, and A. D. Ames, "Robustness of control barrier functions for safety critical control," in *Analysis and Design of Hybrid Systems(Volume 48)*, 2015, pp. 54–61.

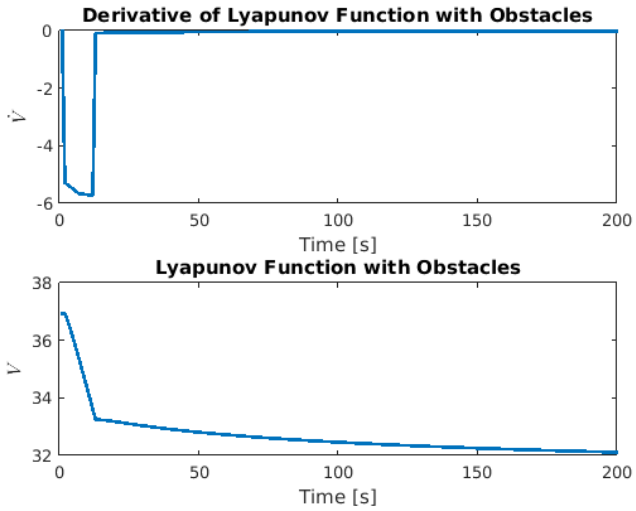


Fig. 8: Lyapunov Function with Obstacles